

爱创课堂前端培训

JS 基础

第 3 天课堂笔记（本课程共 8 天）

班级：北京前端训练营 7 期

日期：2017 年 5 月 12 日

爱创课堂官网：www.icketang.com

目录

JS 基础.....	1
目录.....	1
复习.....	2
一、三元运算符.....	3
二、switch 语句.....	4
三、循环语句.....	7
3.1 for 循环.....	7
3.2 穷举思想.....	10
3.3 累加器.....	11
3.4 累乘器.....	12

复习

数学运算符:

特殊情况: 纯数字字符串参与运算时进行隐式转换。(加号除外)。

特殊字符串 `true`, `false`, `null` 参与运算时也会隐式转换。加法也一样。

`NaN` 参与运算时得到的都是 `NaN`。

1 `true`→1, `false`→0, `null`→0

比较运算符:

特殊情况: 纯数字字符串进行比较时, 也会隐式转换。

特殊字符串 `true`, `false`, `null`。参与比较时也是隐式转换。但是 `null` 进行相等, 或者全等时, `null` 不等于 0;

```

> console.log(null >= 0)
true
< undefined
> console.log(null <=0)
true
< undefined
> console.log(null ==0)
false
< undefined
> console.log(null ===0)
false
< undefined

```

逻辑运算符: &&, ||, !

```

1 false && true    //false
2 true || false   //true
3 !true          //false
4 !false         //true

```

短路语法:

逻辑与运算: 相当于串联电路, 当 **a** 为假时, 输出 **a**。否则的输出 **b**。

逻辑或运算: 相当于并联电路, 当 **a** 为真时, 输出 **a**。否则输出 **b**。

转换为 **true** 情况: 非 0 的数字, 非空的字符串, Infinity

转换为 **false** 情况: NaN , 0 , null , "" , undefined

流程控制语句: **if** 语句。

```

1 if(条件表达式){
2     当满足条件表达式时, 执行的结构体;
3 }else{
4     当不满足条件表达式时, 执行的结构体;
5 }
6 if 语句之后的语句    //都会执行

```

殊途同归: 就是指我们 **if** 语句不管执行那个结构体, **if** 语句执行完毕都会执行 **if** 语句之后的语句。

多分支 **if** 语句:

记住跳楼现象: **if** 语句 **只会执行一个结构体**, 不会重复执行。

一、三元运算符

三元运算符也叫三元表达式。是唯一一个涉及 3 个参数的表达式。

表达式的形式:

```
1 条件表达式 ? 值 1 : 值 2;
```

表示: 当条件表达式为真时, 取值 1;

当条件表达式为假时, 取值 2。

```
1 var a = (7 > 8) ? 8 : 7;
```

```
2 console.log(a);
```

```
7
```

```
1 var a = (7 < 8) ? 8 : 7;
2 console.log(a);
```

```
8
```

三元运算符也是条件分支语句。三元表达式都可以转为 if 语句。等价写法：

```
1 if(7 > 8){
2     console.log(7);
3 }else{
4     console.log(8);
5 }
```

三元表达式有自己优势，当变量的赋值有 2 种可能时，也就是二选一的情况用三元表达式非常简单。

小案例：

```
1 // 计算年终奖。
2 // 当工作年限为不满 1 年，工资小于 8000，年终奖为工资倍数的 1 倍，否则是 1.2 倍；
3 // 当工作年限为不满 2 年，工资小于 10000，年终奖为工资倍数的 1.5 倍，否则是 1.7 倍；
4 // 当工作年限为超过 2 年，工资小于 13000，年终奖为工资倍数的 2.3 倍，否则是 3 倍；
5 if(year == 0){
6     beishu = salary < 8000 ? 1 : 1.2;
7 }else if(year == 1){
8     beishu = salary < 10000 ? 1.5 : 1.7;
9 }else{
10     beishu = salary < 13000 ? 2.3 : 3;
11 }
```

备用选项：我们通常还可以用逻辑运算符，实现备用选项&&，||

```
1 var score = parseInt(prompt("请输入您这次考试的成绩")) || 0;
2 var jie = score < 60 ? alert("不及格") : alert("及格");
```

二、switch 语句

switch 语句也是我们条件分支语句，又叫**开关语句**，允许一个程序去计算一个**表达式的值**，让这个值去匹配一个 **case** 选项,匹配成功直接执行该 **case** 中的**结构体**。

语法：

```
1 switch(expression){
2     case label_1:statements1; //匹配 label_1，执行 case 中的结构体
3     break; //强制跳出该 switch
4     case label_2:statements2; //匹配 label_2，执行 case 中的结构体
5     break; //强制跳出该 switch
6     .....
7     default: statementsx; //表示前面所有的 case 都不满足执行的结构体
8     break; //这个 break 可有可无
9 }
```

①default 语句可以不写，表示前面的 case 都不满足，直接跳出该 switch 语句。

default 后面的 break 可有可无：因为不写 break 也可以直接跳出该 switch。

```
1 var xingzuo = prompt("请输入您的星座");
2 switch(xingzuo){
```

```

3      case "白羊座":
4          alert("对于公司出台的新规定，会感觉受到限制，有抵制情绪产生。不是别人对自己挑剔而是平时自己太任性，约束得不够而已。感情上出现竞争时，不要太过狂妄，赶快进入备战状态吧。对钱财的支出要有一个计划才行，不然难免会出现荷包干涩的状况。");
5          break;
6      case "金牛座":
7          alert("今天与另一半的相处比较融洽，你的好情绪能感染对方，另彼此身心愉悦。财运方面，你的收入水平持续的增长，来自多方面的补益让人羡慕不已。工作上外援助力强，困难之时易得到他人的无私帮助。");
8          break;
9      case "双子座":
10         alert("今天容易受到流言蜚语的中伤，不要太在意，以免因此阻碍你前进的步伐。单身者会感到空虚，不能因此而放纵自己。财运较好，新的进财渠道让你收入颇丰，对需要帮助的人伸出援手会给你带来好运。");
11         break;
12     case "巨蟹座":
13         alert("已婚者享受着被疼爱的感觉，有了情感的滋润，做事的热情尤为浓厚。创业者商机源源不断，关键时刻还能获得意想不到的帮助，即便是难关亦能顺利度过。今天工作可得用心一点，上司在看着你喔！");
14         break;
15     default:
16         alert("没有您输入星座");
17         break;
18 }

```

②switch 语句都可以改成 if 语句。

```

1  if(xingzuo == "白羊座"){
2      alert("对于公司出台的新规定，会感觉受到限制，有抵制情绪产生。不是别人对自己挑剔而是平时自己太任性，约束得不够而已。感情上出现竞争时，不要太过狂妄，赶快进入备战状态吧。对钱财的支出要有一个计划才行，不然难免会出现荷包干涩的状况。");
3  }else if(xingzuo == "金牛座"){
4      alert("今天与另一半的相处比较融洽，你的好情绪能感染对方，另彼此身心愉悦。财运方面，你的收入水平持续的增长，来自多方面的补益让人羡慕不已。工作上外援助力强，困难之时易得到他人的无私帮助。");
5  }else if(xingzuo == "双子座"){
6      alert("今天容易受到流言蜚语的中伤，不要太在意，以免因此阻碍你前进的步伐。单身者会感到空虚，不能因此而放纵自己。财运较好，新的进财渠道让你收入颇丰，对需要帮助的人伸出援手会给你带来好运。");
7  }else if(xingzuo == "巨蟹座"){
8      alert("已婚者享受着被疼爱的感觉，有了情感的滋润，做事的热情尤为浓厚。创业者商机源源不断，关键时刻还能获得意想不到的帮助，即便是难关亦能顺利度过。今天工作可得用心一点，上司在看着你喔！");
9  }else{
10     alert("没有您输入的星座");
11 }

```

③break 表示强制跳出该 switch 语句，如果没有 break，语句在匹配完之后，执行了想要的代码部分，不会自动跳出，会继续往下执行代码，哪怕后面的 case 不匹配，直到遇到一个 break，才会跳出。

```

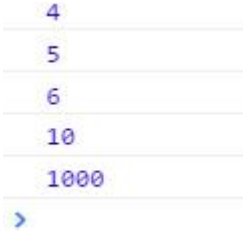
1  var a = 4;
2  switch(a){
3      case 1:console.log(1);
4      case 2:console.log(2);
5      case 3:console.log(3);
6      case 4:console.log(4);
7      case 5:console.log(5);
8      case 6:console.log(6);
9      default:console.log(10);

```

```

10 }
11 console.log(1000);

```



为什么 5,6,10 会输出。因为匹配到了 case 4 后面没有 break 会继续执行,即使不满足后面的 case,也会输出后面的结构体。

虽然 break 必须写,但是有时我们反倒可以利用一下这个特性:

```

1 // 2月28天,1,3,5,7,8,10,12月31,其他的是30天
2 var month = parseInt(prompt("请输入一个月份"));
3 switch(month){
4     case 1:
5     case 3:
6     case 5:
7     case 7:
8     case 8:
9     case 10:
10    case 12:alert("该月有31天!!!!");
11        break;
12    case 2:alert("该月有28天");
13        break;
14    default:alert("该月有30天");
15        break;
16 }

```

错误写法:一下写法是错误的,因为计算机再遇见表达式时,它会先计算表达式。结果 1。

```

17 var month = parseInt(prompt("请输入一个月份"));
18 switch(month){
19     case 1 || 3 || 5 || 7 || 8 || 10 || 12:
20         alert("该月有31天!!!!");
21         break;
22     case 2:alert("该月有28天");
23         break;
24     default:alert("该月有30天");
25         break;
26 }

```

switch 还有一种另类写法:直接在 switch 中写 true,在 case 语句中书写表达式。当 case 中的表达式为真时,可以与 switch 语句中的 true 匹配输出该 case 中的结构体。

```

1 var shu = parseInt(prompt("请输入您的分数")) || 0;
2 switch(true){
3     case shu < 60:
4         alert("不及格");
5         break;
6     case shu < 80:
7         alert("良好");
8         break;
9     case shu < 90:
10        alert("优秀");
11        break;
12    default:
13        alert("非常棒!!!");

```

14 }

该另类写法，语法上也不是特别清晰，还不如 `if` 语句。

总结：以上三种条件分支语句适用的情况

`if` 语句是最常用的，用途最广泛。一定要牢牢掌握。

三元表达式适用于变量的赋值是二选一的情况时，最适合。

`switch` 语句是当一个值去匹配多种情况时，最适合。

三、循环语句

循环语句就是重复执行一段代码，当不满足某个条件时，结束循环。

循环语句：for 循环，do while 循环，while 循环，for in 循环。

3.1 for 循环

for 循环是前置判断。就是先判断条件表达式，满足条件表达式，就执行。

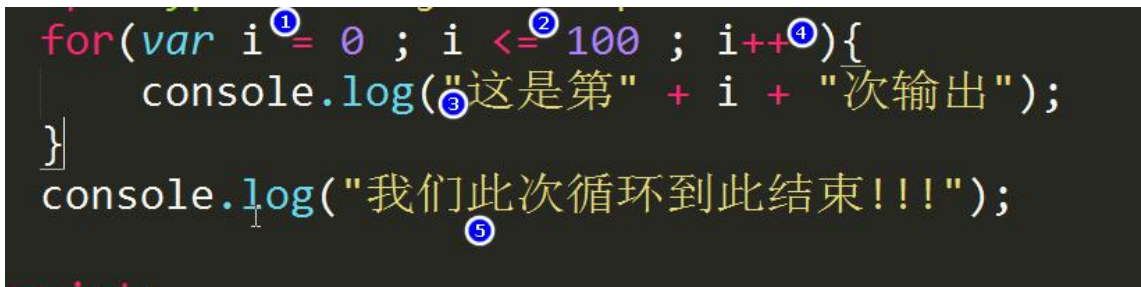
for 循环重复执行一段代码，直到我们条件为假的时候，跳出循环。

```
1 for(初始化变量;最大值;步长){
2     循环语句;
3 }
```

```
1 for(var i = 0 ; i <= 100 ; i++ ){
2     console.log("这是第" + i + "次输出");
3 }
```

```
这是第94次输出
这是第95次输出
这是第96次输出
这是第97次输出
这是第98次输出
这是第99次输出
这是第100次输出
```

for 循环的顺序：



```
for(var i① = 0 ; i② <= 100 ; i④++){
    console.log③("这是第" + i + "次输出");
}
console.log⑤("我们此次循环到此结束!!!");
```

for 循环先执行步骤 1，且执行一次。进行步骤 2 条件语句的判断，当结果为假，直接跳出 for 循环，执行语句 5；

for 循环先执行步骤 1，且执行一次。当判断结果为真时，执行语句 3（也就是我们的循环语句），然后执行语句 4。然后再次进行语句 2 的判断，当结果为假时，跳出 for 循环。执行语句 5。

for 循环先执行步骤 1，且执行一次。当判断结果为真时，执行语句 3（也就是我们的循环语句），然后执行语句 4。然后再次进行语句 2 的判断，当结果为真时，执行语句 3，然后一直循环语句 3，语句 4，语句 2……一直循环到语句 2 为假时结束 for 循环，执行语句 5。

我们一定要明白 for 循环的机理，可以不用直接输出，自己进行预判断。

```
1 // 案例一
2 for(var i = 4 ; i <= 14 ; i += 3){
3     console.log(i);
4 }
```

4
7
10
13

```
1 // 案例二
2 for(var i = 3; i < 13 ; i += 4){
3     console.log(i++);
4 }
```

3
8

我们在写 for 循环时，常会用到一个变量，这个变量就叫循环变量。循环变量的名称可以随意定义，通常我们使用 i,j,k。循环变量是全局变量。可以放在 for 循环外。

错误写法：

```
1 var i = 3;
2 for(; i < 10 ; i += 2 ;){
3     //前两个分号必须写，即使没有变量的初始化也要写!!!
4     //语句最后的分号千万千万千万不要写。
5 }
```

正确写法：

```
1 // 案例三
2 var i = 3;
3 for(; i < 10 ; i += 2){
4     console.log(i);
5 }
```

3
5
7
9
>

```
1 // 案例四，步长可以减少
2 for(var i = 12 ; i > 2 ; i -= 4){
3     console.log(i);
4 }
```

12
8
4

```
1 // 案例五
2 for(var i = 4 ; i != 10 ; i += 3){
3     console.log(i);
4 }
```



```
4
7
```

```
1 // 案例六
2 var i = 3;
3 for(console.log(i++); i < 13 ; i += 5){
4     console.log(i);
5 }
```

```
3
4
9
```

```
1 // 案例七
2 var i = 3;
3 for(console.log(i++); i > 13 ; i += 5){
4     console.log(i);    //一次都不执行
5 }
```

```
3
```

死循环：就是一直让判断为真，不会跳出循环。

```
1 // 案例八，死循环
2 for(var i = 1 ; i > 0 ; i++){
3     console.log("哈哈");
4 }
```

for 循环语句可以嵌套 for 循环语句。（for 循环的嵌套，想执行循环语句不但要满足外层的 for 循环还要满足内层的 for 循环。）

注意：这两的 for 循环语句中的变量名不要相同。

```
1 // 案例九 for 循环嵌套 for 循环
2 for(var i = 1 ; i < 5 ; i++){
3     for(var j = 1 ; j < 5 ; j ++){
4         console.log(i , j);
5     }
6 }
```

```

1 1
1 2
1 3
1 4
2 1
2 2
2 3
2 4
3 1
3 2
3 3
3 4
4 1
4 2
4 3
4 4

```

for 循环语句还可以嵌套 if 语句。（想要执行 if 语句中的结构体，必须先满足 for 循环的条件还要满足内部的 if 语句的判断条件）

```

1 // 案例十 for 循环语句嵌套 if 语句
2 for(var i = 4 ; i < 20 ; i += 5){
3     if(i % 2 == 0){
4         console.log(i);
5     }
6 }

```

```

4
14
>

```

3.2 穷举思想

概念：我们经常想得到一组数据，有一些特定的要求，计算机没法自己帮我们输出这些数据。我们人为的需要编写一段程序，让计算机去帮我们实现程序。将所有的可能情况，一一的列举出来，然后我们人为限定判断条件，把符合条件的数据就给它输出，不满足的就跳过，继续验证下一个数据是否满足条件，直到把所有可能情况都验证一个遍。这个方法就叫做全举法，也叫穷举法。

外层：一一列举，for 循环。

内层：进行判断，if 语句。

案例 1：得到某个数的所有的约数。

$a \% b == 0$ 此时，a 叫做 b 倍数，b 叫做 a 的约数也叫因数。

任何一个数的约数最小是 1，最大约数是他本身。

```

1 var num = parseInt(prompt("请输入一个正整数"));
2 for(var i = 1 ; i <= num ; i++){
3     if(num % i == 0){
4         console.log(i);
5     }
6 }

```

```

1
2
3
4
6
12

```

案例 2: 找出所有 3 位正整数间的水仙花数。

条件 1: 是 3 位数 100-999 之间的数。

条件 2: 各个位数的立方数之和等于原数据。

$$155 = 1^3 + 5^3 + 5^3$$

3.3 累加器

有时候我们并不想列举出所有符合条件的数据, 就想得到一个总数。这时我们需要用到一个累加器。

本身是一个变量, 可以随时自加数。

找到一个符合条件的数据, 就给变量加 1。

最终循环完之后, 得到了一个总数。

注意: ①累加器定义必须在循环外面。②初始值必须是 0。③看总数的时候, 必须是在循环结束后, 在外面看总数。

案例: 想得到任意一个正整数约数的总个数。

```

1 // 输入数字
2 var num = parseInt(prompt("请输入一个正整数"));
3 // 在 for 循环外面定义累加器
4 var sum = 0;
5 // 寻找约数, 当满足条件时, 累加器自加 1
6 for(var i = 1 ; i <= num ; i ++){
7     // 求约数
8     if(num % i == 0){
9         // 满足条件时, sum+1
10        sum ++;
11        console.log(i);
12    }
13 }
14 // 跳出 for 循环在外面看 sum 总值
15 console.log(num + "的约数总个数是" + sum );

```

```

1
2
4
8
8的约数总个数是4
> |

```

累加升级版:

用户输入一个数字 n

计算 $\frac{3}{2} + \frac{4}{3} + \frac{5}{4} + \dots + \frac{n+1}{n}$

```

1 // 输入一个正整数
2 // 累加是(n+1)/n
3 var n = parseInt(prompt("请输入一个正整数"));
4 // 累加器
5 var sum = 0;
6 // for 循环
7 for(var i = n ; i >= 2 ; i --){
8     // 累加器
9     sum += (n + 1) / n;
10 }
11 // 跳出循环输出 sum
12 console.log(sum);

```

3.4 累乘器

累乘器：保存我们前面乘数的积。

阶乘： $8! = 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1$

注意：

- ①必须写在循环外面。
- ②初始值必须是 1.
- ③输出也必须是在 for 循环完毕之后。

```

1 // 用户输入
2 var num = parseInt(prompt("请输入一个正整数"));
3 // 在 for 循环外定义累成器,初始值为 1
4 var cheng = 1;
5 // 用 for 循环穷举
6 for(var i = 1; i <= num ; i ++){
7     cheng *= i;
8 }
9 // 累乘器在 for 循环外输出结果
10 console.log(num + "的阶乘是" + cheng);

```

1

1

1

1

1

1

1

1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1